Titel

[Abkürzungsverzeichnis 3](#_Toc104386689)

[RS-485 4](#_Toc104386690)

[Modbus 7](#_Toc104386691)

[Verzeichnisse 13](#_Toc104386692)

[Quellenverzeichnis 13](#_Toc104386693)

[Abbildungsverzeichnis 14](#_Toc104386694)

[Tabellenverzeichnis 15](#_Toc104386695)

## Abkürzungsverzeichnis

PLC: Programmable Logic Controller

TCP: Transmission Control Protocol

IP: Internet Protocol

TCP/IP: Transmission Control Protocol/Internet Protocol (Internet Protocol Suite)

RTU: Remote Terminal Unit

ASCII: American Standard Code For Information Interchange

ADU: Application Data Unit

PDU: Protocol Data Unit

Dec.: Dezimal

Hex.: Hexadezimal

Bin.: Binär

RS: Recommended Standard

TIA/EIA: Telecommunications Industry Association/Electronic Industrie Alliance

ISO/IEC: International Organization for Standardization/International Electrotechnical Commission

PC: Personal Computer

V: Volt

Abb.: Abbildung

bps: Bit pro Sekunde

CRC: Cyclic Redundency Check

RJ45: Registered Jack 45 (Steckverbindung)

D-Sub (9): D-Subminiature mit 9 Kontakten (Steckverbindung)

OSI: Open Systems Interconnection Model

8-N-1: Startbit - 8x Datenbit – Stoppbit (Bitübertragungsformat)

7-E-1: Startbit - 7x Datenbit – Even Paritätsbit - Stoppbit (Bitübertragungsformat)

PS/2: Benannt nach dem 1987 IBM Personal System/2 Computer, Schnittstelle für Eingabegeräte, wie Maus, Tastatur oder Trackball

UART: Universal Asynchronous Receiver-Transmitter

FIFO: First In, First Out (Speicherprinzip)

## Serielle Kommunikation

Wenn zwei Parteien miteinander kommunizieren wollen, müssen sie sich vorher auf eine Kommunikationsart einigen, damit jede Nachricht sein Ziel erreicht und dort auch verstanden wird. Serielle Kommunikation ist eine sehr simple Kommunikationsart, da nur ein Leiter benötigt wird, welcher an und aus geschaltet wird. Damit Datenbits ausgelesen werden können, wird eine Clock oder eine andere Art an Zeitreferenz benötigt. Bei einer synchronen Verbindung gibt es eine zentrale Clock, die allen Geräten die Zeit vorgibt. Problem bei synchronen Verbindungen ist, dass alle kommunizierende Geräte mindestens einmal nach dem Einschalten das Signal der Clock empfangen haben müssen um sich zu Synchronisieren. Dies ist in vielen Anwendungsfälle nur schwer oder gar nicht möglich [1, S. 11–13].

Bei asynchronen Verbindungen wird keine Clock benötigt, da bei jeder Nachricht zwischen den Geräten diese selbst erkennen, wie die Nachricht auszulesen ist. Es gibt viele verschiedene Formate für das asynchrone Nachrichtenübertragen, ein Beliebtes aber ist 8-N-1. Dort folgen auf ein Startbit acht Datenbits und ein Stoppbit. Das N besagt, dass es kein Paritätsbit gibt. Parität sagt aus, ob es eine gerade oder ungerade Anzahl an 1 in den Datenbits gibt. Wenn bei einer empfangenen Nachricht das Paritätsbit nicht mit der Parität der Datenbits übereinstimmt, so wird die Nachricht verworfen, da es mindestens einen Fehler gibt. Ein Even Paritätsbit wird 1, wenn die Datenbits eine gerade Anzahl an 0 enthält und ein Odd Paritätsbit wird 1, wenn die Nachricht eine ungerade Anzahl an 0 enthält. Die meisten seriellen Schnittstellen unterstützen zwischen einem Start- und einem Stoppbit 5-8 Datenbits sowie ein Paritätsbit [1, S. 13–14].

Die Bitübertragungsrate einer Verbindung wird in Bits pro Sekunde angegeben. Die Datenänderungsübertragungsrate, auch Baud-Rate genannt, ist in vielen Fällen gleich der Bitübertragungsrate, da bei seriellen Verbindungen jedes Bit eine Datenänderung ist. Bei Telefon- oder Ethernetverbindungen jedoch ist die Baud-Rate um einiges niedriger als die Bitübertragungsrate, da in jede Datenänderung mehrere Bits kodiert sind. Das Hinzufügen eines Start- und Stoppbits bei 7-E-1 verringert die Übertragungsgeschwindigkeit um 30%, da auf sieben Datenbits zwei Formatbits und ein Paritätsbit kommen. Bei manchen Übertragungsprotokollen wird das Stoppbit doppelt so lang wie ein Datenbit gesetzt oder mehrere Paritätsbits verwendet um die Fehlerquote zu verringern. Ob die Verringerung der Datenübertragungsrate die erhöhte Fehlerreduktion wert ist, muss von Anwendungsfall zu Anwendungsfall entschieden werden [1, S. 13–15].

Serielle Anschlüsse waren für Jahrzehnte die bevorzugte Art um externe Geräte mit einem Computer zu verbinden. Aber bis auf ein oder zwei synchrone, serielle PS/2-Schnittstellen um Tastatur und Maus anzuschließen [2] werden keine serielle Anschlüsse mehr in modernen Computer verwendet. Serielle Schnittstellen sind und werden aber weiterhin in Kontroll- und Überwachungssystem eingesetzt, da sie günstig, wenig fehleranfällig und einfach zu bedienen sind [1, S. 26].

### UART

Ein Universal Asynchronous Receiver-Transmitter, UART genannt, ist die Schnittstelle zwischen dem parallelen internen Bussystem eines (Mikro)Computers und einem seriellen Übertragungssystem. Dieser wandelt die zu sendende Daten des Computers in ein serielles Format um und wandelt die empfangenen seriellen Daten in ein Format um, welches für den Computer lesbar ist. Außerdem sind viele UART für das Management der seriellen Verbindung zuständig, wie das Übertragungstiming verschiedener Geräte am selben Bus oder das automatische Anpassen an dieselbe Baud-Rate zwischen verschiedenen Geräten [1, S. 26].

In einem UART sind Buffer enthalten, die mehrere Byte zwischenspeichern können. So muss der Computer sich nicht um jeden einzelnen empfangen Bit kümmern, sondern kann die empfangene Nachricht in einem Zug auslesen. Dies ist auch von Vorteil, da das computer-interne Bussystem um ein Vielfaches schneller ist, als eine serielle Leitung. Jenes gilt ebenso beim Senden von seriellen Nachrichten. Der Computer kann die komplette zu sendende Nachricht an den UART senden, wo diese im Buffer gespeichert und daraufhin Stück für Stück gesendet wird. Diese Buffer sind first-In, first-out, kurz FIFO. Es wird also der zuerst empfangene Bit als erstes gesendet [1, S. 27]. Des Weiteren hat ein UART verschiedene Speicherregister. Unteranderem jeweils eines für die als letztes empfangenen und gesendeten Bits. Außerdem sind dort Konfigurationen, Statusinformationen, Bufferbenutzung und Fehlermeldungen gespeichert [1, S. 40–42].

### RS-485

RS-485 ist ein Datenübertragungsstandard, wurde von der Telecommunications Industry Association und der Electronic Industrie Alliance entwickelt und ist auch unter dem Namen TIA/EIA-485 und ISO/IEC 8482.1993 bekannt [1, S. 185]. RS-485 wird in Industrie-, Medizin- und Konsum-Elektronik eingesetzt. Dieser Standard liegt auf OSI-Layer 1 und definiert nur die elektrische Hardware der Sender, Empfänger und Übertragungsmedien. Es ist möglich eine RS-485 Implementierung in halb oder voll Duplex zu betreiben, bei voll Duplex werden aber statt zwei vier Leiter benötigt. Außerdem sollte RS-485 in einer Bus-Topologie betrieben werden mit einem Haupt-Bus, von dem kurze Abzweigungen zu den einzelnen Sendern und Empfängern gehen [3, S. 1–2].

RS-485 benutzt ausgeglichene Übertragungsleitungen, wo jede Verbindung aus zwei Leitern (A und B) besteht und in jeder Leitung jeweils die diametrale Spannung zur anderen Leitung herrscht. Der Empfänger reagiert auf die Spannungsdifferenzen der beiden Leitungen und diese Übertragungsart wird deshalb auch Differentialsignalisierung genannt. Ein großer Vorteil von einem solchen Leiteraufbau ist die starke Reduzierung von Rauschen, da die Felder der einzelnen Leiter sich gegenseitig auslöschen. Außerdem sind die Leiter so relativ immun gegen Unterschiede im Erdpotential bei längeren Übertragungsstrecken [1, S. 186–187].

Ein RS-485-Sender benötigt mindestens eine 1,5V Differenz zwischen seinen beiden Ausgängen. Die Differenz zur Erde ist nicht definiert, darf aber nicht mehr als 7 Volt von der Erde abweichen. Bei dem Empfänger muss die Differenz zwischen den zwei Leitern immer noch mindestens 0,2V betragen um korrekt erkannt zu werden. Wenn Leiter A mindestens 0,2V größer als Leiter B ist, wird eine 1 registriert und wenn Leiter A mindestens 0,2V kleiner als Leiter B ist, wird eine 0 registriert. Das heißt, dass auf dem Weg zwischen Sender und Empfänger mindestens 1,3V an Rauschen zu dem Signal hinzukommen können, es aber trotzdem erkannt wird [1, S. 190–191].

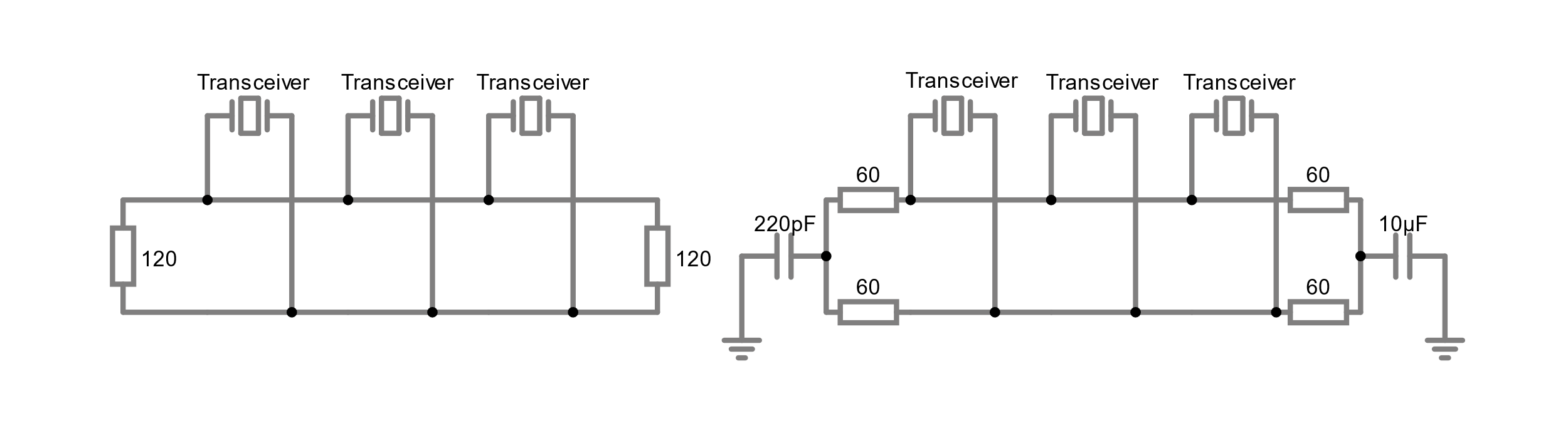
Wie in Abb. 1 zu sehen ist, müssen die Enden der beiden Haupt-Bus-Kabeln mit der, den Kabeln entsprechenden, charakteristischen Impedanz abgeschlossen werden, um Reflektionen vorzubeugen. Da der RS-485 Standard Kabel mit einer charakteristischen Impedanz von 120 Ohm vorschlägt, sind die meisten Abschlusswiderstände auch 120 Ohm stark. Wenn eine Verbindung in einem Umfeld mit starken elektrischen Interferenzen gelegt werden soll, werden die 120 Ohm Abschlusswiderstände meist mit zwei 60 Ohm Widerständen und einem Tiefpassfilter dazwischen ersetzt [3, S. 3]. 

Abb. RS-485 Abschlusswiderstände [3, S. 3]

Eine RS-485-Verbindung kann entweder bis zu 10 Mbps schnell oder bis zu 1200 Meter lang sein. Wie man in Abb. 2 sieht, steigt die Übertragungsrate ab 100 Kbps aufgrund der geringeren Kapazität des Kabels bei kürzerer Länge [1, S. 193].

Abb. RS-485 Geschwindigkeit in Abhängigkeit der Kabellänge [1, S. 193]

### Modbus

Modbus ist ein Datenübertragungsprotokoll, welches von Modicon, jetzt Schneider Electric, im Jahr 1979 entwickelt wurde. Es wird von hunderten verschiedenen Herstellern in tausenden verschiedenen Geräten eingesetzt und ist damit der de facto Standard um PLCs, Computer, Sensoren und Aktoren miteinander zu verbinden [4, S. 508], [5]. Es gibt verschiedene Versionen von Modbus, unter anderem Modbus RTU, Modbus ASCII, Modbus über TCP/IP, Modbus Plus und Modbus Enron [4, S. 509], [5]–[7]. Serieller Modbus (Modbus RTU, Modbus ASCII,) beruht auf dem Master/Slave-System, wo eine einziges Master-System bis zu 247 Slave-Systeme steuern kann [4, S. 508]. Die folgenden Kapitel behandeln nur Modbus RTU.

Das Master-System kann zwei Arten von Nachrichten (Requests) an Slave-Systeme schicken: Unicast, wo das Master-System genau ein Slave-System anspricht, welches einen Prozess ausführt und ihm danach mit einer Reply antwortet, und Broadcast, wo das Master-System allen Slave-Systemen eine Request schickt, welche aber nur ein Schreibbefehl sein kann. Bei einem Broadcast wird keine Reply von den Slaves übermittelt.

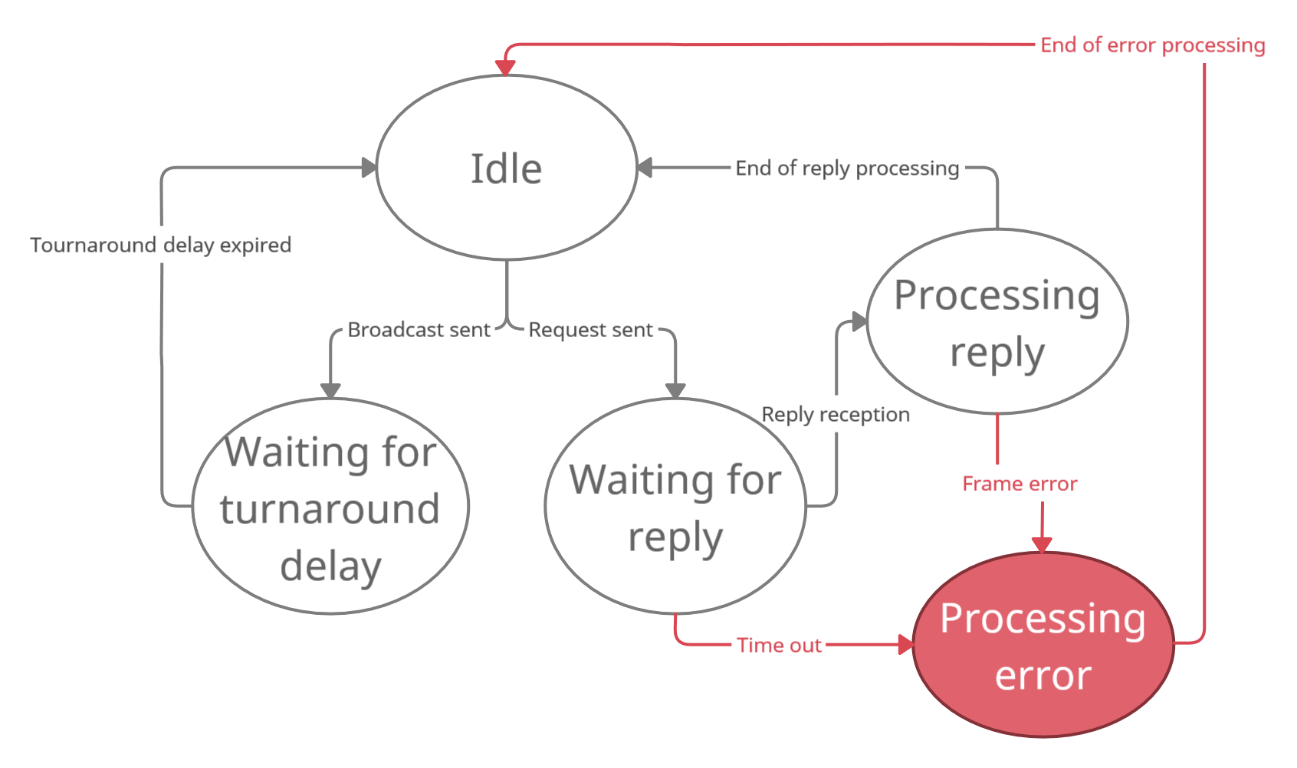


Abb. Modbus Master Zustandsdiagramm [8, S. 9]

In Abb. 3 ist der vereinfachte Prozessablauf eines Modbus-Master-Systems dargestellt. Dieser empfängt von einem im OSI-Modell höher liegenden Protokoll Befehle. Das Master-System ist ohne Auftrag von außen im Ruhemodus (Idle). Wenn nun ein Protokoll einen Prozess in einem oder mehreren Slaves ausführen will, sendet es einen Befehl an das Master-System. Wenn es ein Broadcast-Befehl ist, sendet der Master diesen an alle Slaves und wartet dann eine vorher definierte Zeit (Turnaround delay) bevor es wieder in den Ruhemodus zurück geht. Bei einem Unicast-Befehl sendet der Master diesen an das ausgewählte Slave-System und wartet auf eine Reply. Wenn diese nicht nach einer im Voraus festgelegten Zeit zurückkommt oder diese empfangen wurde, aber Fehler beinhaltet, geht der Master zurück in den Ruhemodus. Wenn eine fehlerfreie Reply innerhalb des festgelegten Zeitfenster empfangen wird, wird diese an das auftraggebende Protokoll gegeben [8, S. 9–10].

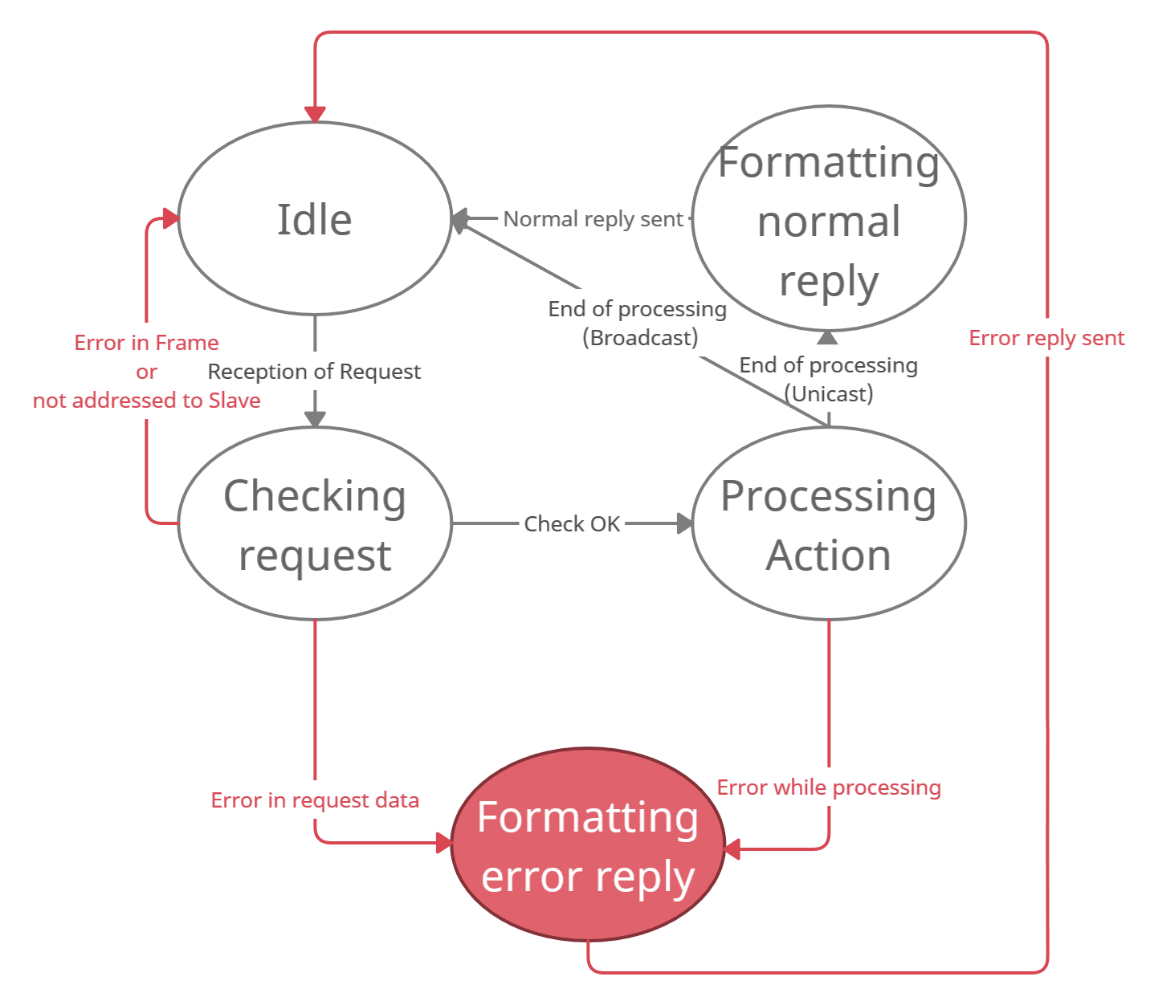


Abb. Modbus Slave Zustandsdiagramm [8, S. 10]

In Abb. 4 ist der komplette Prozessablauf eines Modbus Slaves dargestellt. Beim Starten des Slaves-Systems befindet sich dieses im Ruhemodus (Idle). Wenn der Slave eine Nachricht empfängt, wird diese zuerst auf das korrekte Format überprüft. Außerdem stellt der Slave fest, ob die Nachricht an seine Adresse gerichtet ist. Wenn die Nachricht förmliche Fehler oder die falsche Adresse enthält, wird diese Nachricht ignoriert und der Slave geht zurück in den Ruhemodus. Ist die Nachricht an den Slave adressiert und förmlich korrekt, wird zuerst überprüft ob er den Befehl in dieser Nachricht ausführen kann. Danach wird der Befehl ausgeführt. Wenn bei diesen beiden Prozessen Fehler vorkommen, wird eine Fehlernachricht (error reply) an den Master geschickt, die einem dem Fehler entsprechenden Code enthält. Gibt es keine Fehler, geht der Slave bei einer Broadcast-Nachricht direkt nach der Ausführung, und bei eine Unicast-Nachricht nach dem Senden der Antwort (Reply), in den Ruhemodus [8, S. 10].

Jedes übertragene Daten-Byte in einer Modbus-Nachricht enthält acht Datenbits, beziehungsweise zwei hexadezimal Zeichen, und 3 Formatbits, das Start-, Stopp-, und Paritätsbit. Im Default-Paritätsmodus wird bei einer ungeraden Anzahl an Einsen das Paritätsbit auf eins gesetzt. In Abb. 5 ist zu sehen, dass zuerst das Startbit, dann die Datenbits und danach das Paritäts- und das Stoppbit. Außerdem ist es möglich das Paritätsbit durch ein zweites Stoppbit zu ersetzen [8, S. 12].

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start | Data | Data | Data | Data | Data | Data | Data | Data | Par | Stop |

Abb. Bitsequenz Modbus mit Paritätsbit [8, S. 12]

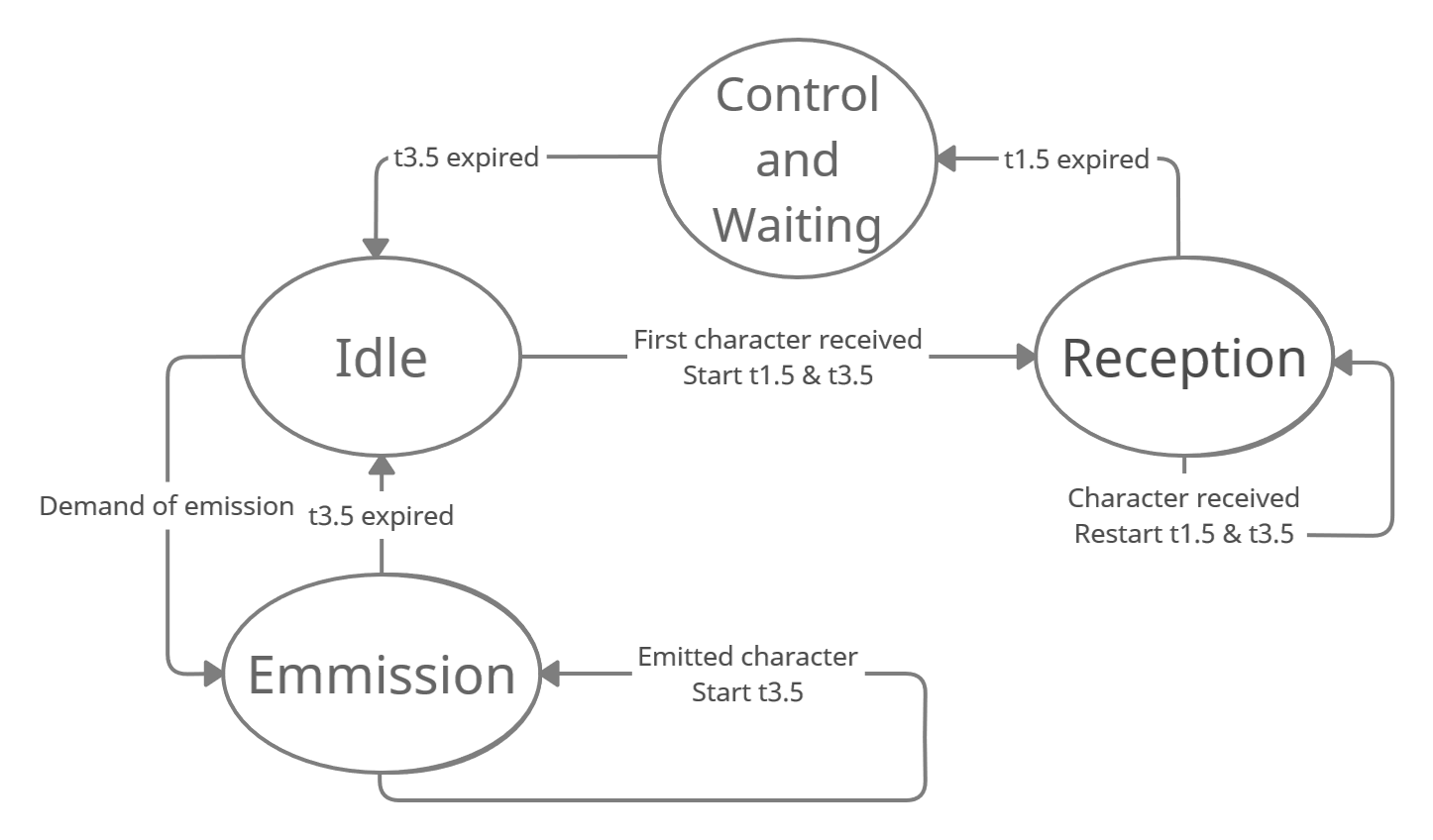


Abb. Modbus Übertragung Zustandsdiagramm [8, S. 14]

In Abb. 6 sind die verschiedenen Zustände eines Modbus-Senders/Empfängers dargestellt. Diese sind bei Master und Slave gleich. Zwischen zwei Modbus-Nachrichten müssen mindestens 3,5 zeichenlang Stille sein und wenn zwischen 2 Zeichen mehr als 1,5 zeichenlang Stille ist, wird die Nachricht vom Empfänger als beendet betracht. Nach dem Empfangen eines Zeichens starten beide Timer (t1.5 & t3.5) und werden beim Empfangen des nächsten Zeichen wieder neugestartet. Nachdem der 1,5 Zeichen Timer überschritten wird, wird bei der empfangenen Nachricht die Controlframes (Parität, CRC und richtige Adresse) überprüft. [8, S. 13].

Eine vollständige Nachricht im Modbus-Protokoll ist in Abb. 7 zu sehen. Diese wird Application Data Unit (ADU) genannt und besteht aus einer Adresse, der Protocol Data Unit (PDU), bestehend aus dem Funktionscode und den zu übertragenden Daten, und schlussendlich dem Fehlercheck. Eine Nachricht kann maximal 256 Byte groß sein [8, S. 13].

|  |  |  |  |
| --- | --- | --- | --- |
| ADU | | | |
|  | PDU | |  |
| Slave Adresse | Funktionscode | Daten | CRC |
| 1 Byte | 1 Byte | 0 bis 252 Byte | 2 Byte |

Abb. Modbus Frame [8, S. 13]

Es gibt 247 valide Slave-Adressen und eine Broadcastadresse. Im Funktionscodefeld steht eine acht Bit lange Nachricht, die dem Slave sagt, welchen Befehl er auszuführen hat. Es gibt elf verschiedene Funktionscodes, welche alle Modbusübertragungsmedien unterstützen. Jene sind in TABELLE I dargestellt sind. Außerdem gibt es mehrere dutzende Funktionscode, welche Übertragungsmedium spezifisch sind. Desweiteren ist es möglich, dass Gerätehersteller eigene Modbusbefehle implementieren. Das Datenfeld ist n\*8 Bit lang und enthält die zu übertragenden Daten. Als letztes kommt das 16-bit-lange Fehlercheckfeld, wo mit Hilfe von Cyclical-Redundancy-Check-Rechnungen die empfangene Nachricht überprüft werden kann [9, S. 18–25].

TABELLE ALLGEMEINE MODBUS FUNKTIONEN [9, S. 31–66]

|  |  |  |
| --- | --- | --- |
| Code in Dec. | Name | Funktion |
| 01 | Read Coil Status | Liest den AN/AUS Status eines Coils |
| 02 | Read Input Status | Liest den AN/AUS Status eines Inputs |
| 03 | Read Holding Register | Liest die Werte von Holding Registern |
| 04 | Read Input Register | Liest die Werte von Input Registern |
| 05 | Force Single Coil | Setzt den AN/AUS Status eines Coils |
| 06 | Preset Single Register | Setzt die Werte von einem Holding Register |
| 07 | Read Exception Status | Liest den Status von prädefinierten Status Coils. Bspw.: Maschine in Betrieb |
| 08 | Diagnostic | Mehrere Unterfunktionen für verschieden Status von Slaves |
| 15 | Force Multiple Coils | Setzt den AN/AUS Status mehrere Coils |
| 16 | Preset Multiple Register | Setzt die Werte von mehreren Holding Register |
| 17 | Report Slave ID | Zeigt die Slave ID an |



Abb. Beispiel Modbus Funktion 03[9, S. 15]

In Abb. 8 kann man eine Beispielsregisterabfrage sehen. Es wurde nur die PDU dargestellt, bei der gesamten Nachricht wurde am Anfang noch die Adresse und am Ende die CRC-Summe übermittelt. Alle Registeradressen und Registerwerte werden in einen High- und Lowteil aufgeteilt, welche hintereinander gesetzt werden. Der Master fragt dort die Daten von Register 0x6B, 0x6C und 0x6D ab, indem das abzufragende Startregister und die Anzahl der darauffolgenden Register definiert werden. Der Slave antwortet daraufhin erst mit der Funktion und der Byteanzahl der Antwortwerte. Daraufhin folgen die Werte der abgefragten Register. Der Master kann nun die Antwort (Register 0x6B = 0x022B, Register 0x6C = 0x000 und Register 0x6D = 0x0064) an das höherrangige Protokoll senden.

Der Standard Modbus-Datenübertragungsstandard ist RS485 und alle Systeme sind parallel angeschlossen. Es werden mindestens zwei Datenübertragungskabel benötigt, idealerweise aber auch ein Erdkabel. Diese Kabel werden entweder mit Schraub-, RJ45- oder D-Sub(9)-Steckverbinder ausgestattet. Die Standardübertragungsgeschwindigkeit beträgt 19200 bps. Es können aber auch Andere benutzt werden, wie 1200 bps, 9600 bps oder 115000 bps [8, S. 20]. Bei einem RS485-Modbus-Netzwerk können maximal 32 Systeme ohne Repeater angeschlossen werden und werden alle parallel an einen zentralen Bus angeschlossen. Bei einem 9600 bps und 0,2 mm dicken Kabel liegt die maximale Länge des Buses bei 1000 Metern. Abzweigungen von diesem dürfen nicht länger als 20 Meter sein. Damit es zu keinen Reflexionen am Ende des Buses kommt, müssen dort die beiden Signalkabel mit ein 150 Ohm Abschlusswiderstand oder jeweils ein 1 nF Kondensator und ein 120 Ohm Abschlusswiderstand verbunden sein [8, S. 27–28].

# Verzeichnisse

## Quellenverzeichnis

[1] J. Axelson, *Serial Port Complete: Programming and Circuits for RS-232 and RS-485 Links and Networks.* Madison: Lakeview Research, 2002.

[2] Adam Chapweske, „StuBS: The PS/2 Mouse/Keyboard Protocol“. https://www.sra.uni-hannover.de/Lehre/WS21/L\_BST/doc/ps2.html (zugegriffen 21. Juni 2022).

[3] Thomas Kugelstadt, „Texas Instruments: The RS-485 Design Guide“. Mai 2021. Zugegriffen: 20. Juni 2022. [Online]. Verfügbar unter: https://www.ti.com/lit/an/slla272d/slla272d.pdf

[4] B. Drury, *The control techniques drives and controls handbook*, 2nd ed. Stevenage: Institution of Engineering and Technology, 2009.

[5] „Modbus FAQ“. https://modbus.org/faq.php (zugegriffen 5. Mai 2022).

[6] „About Modbus | Simply Modbus Software“. http://www.simplymodbus.ca/FAQ.htm (zugegriffen 5. Mai 2022).

[7] „Modbus Plus | Schneider Electric USA“. https://www.se.com/us/en/product-range/576-modbus-plus/ (zugegriffen 5. Mai 2022).

[8] Modbus Organization, Inc, „MODBUS over Serial Line Specification and Implementation Guide V1.02“. 20. Dezember 2006. Zugegriffen: 20. Mai 2022. [Online]. Verfügbar unter: https://modbus.org/docs/Modbus\_over\_serial\_line\_V1\_02.pdf

[9] MODICON, Inc., Industrial Automation Systems, „Modicon Modbus Protocol Reference Guide“. Juni 1996.

## Abbildungsverzeichnis

[Abb. 1 RS-485 Geschwindigkeit in Abhängigkeit der Kabellänge [1, S. 193] 7](#_Toc106372499)

[Abb. 2 Modbus Master Zustandsdiagramm [6, S. 9] 8](#_Toc106372500)

[Abb. 3 Modbus Slave Zustandsdiagramm [6, S. 10] 9](#_Toc106372501)

[Abb. 4 Bitsequenz Modbus mit Paritätsbit [6, S. 12] 10](#_Toc106372502)

[Abb. 5 Modbus Übertragung Zustandsdiagramm [6, S. 14] 10](#_Toc106372503)

[Abb. 6 Modbus Frame [6, S. 13] 10](#_Toc106372504)

[Abb. 7 Beispiel Modbus Funktion 03[7, S. 15] 11](#_Toc106372505)

## Tabellenverzeichnis

[TABELLE I ALLGEMEINE MODBUS FUNKTIONEN [7, S. 31–66] 11](#_Toc102742126)